Generic DPA attacks: curse or blessing?

Oscar Reparaz, Benedikt Gierlichs, and Ingrid Verbauwhede

KU Leuven Dept. Electrical Engineering-ESAT/COSIC and iMinds
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
firstname.lastname@esat.kuleuven.be

Abstract. Generic DPA attacks, such as MIA, have been recently proposed as a method to mount DPA attacks without the need for possibly restrictive assumptions on the leakage behaviour. Previous work identified some shortcomings of generic DPA attacks when attacking injective targets (such as the AES Sbox output). In this paper, we focus on that particular property of generic DPA attacks and explain limitations, workarounds and advantages. Firstly we show that the original fix to address this issue (consisting of dropping bits on predictions to destroy the injectivity) works in practice. Secondly, we describe how a determined attacker can circumvent the issue of attacking injective targets and mount a generic attack on the AES using previously mentioned non-injective targets. Thirdly, we explain important and attractive properties of generic attacks, such as being effective under any leakage behaviour. Consequently, we are able to recover keys even if the attacker only observes an encrypted version of the leakage, for instance when a device is using bus encryption with a constant key. The same property also allows to mount attacks on later rounds of the AES with a reduced number of key hypotheses compared to classical DPA. All main observations are supported by experimental results, when possible on real measurements.

Keywords: DPA, generic DPA, MIA, KSA

1 Introduction

Side-channel attacks pose nowadays a significant threat for the security of cryptographic embedded devices. Since the first publication [8] in 1996, a respectable body of academic research on side-channel attacks and countermeasures has been produced. Besides, side-channel attacks are not only a topic of academic research but also relevant to industry. Present embedded security devices, such as bank cards, phone SIMs and electronic passports normally feature some kind of protection against side-channel attacks.

An important class of side-channel attacks are Differential Power Analysis (DPA) attacks, introduced in [9] by Kocher et al. They showed that cryptographic keys could be extracted from embedded devices by first measuring their instantaneous power consumption while performing cryptographic operations and subsequently performing a statistical analysis of the power consumption

measurements. The relatively inexpensive equipment needed for performing DPA attacks greatly increases the threat of this family of attacks.

Since [9], a substantial work on the so-called distinguisher in DPA attacks has been carried out [2,4,6,13,10]. In a nutshell, this trend tries to optimize the performance of a DPA attack, measured in the number of traces needed for a successful attack, under the assumption that the device leaks information in a specific manner, usually Hamming weight (HW) or Hamming distance (HD). Pearson's sample correlation coefficient distinguisher is a popular choice when mounting DPA attacks (also known as CPA attacks) on devices whose behaviour can be linearly approximated by some leakage model (usually HD or HW) given beforehand (essentially, acquired based on previous knowledge of the chip and its implementation).

The choice of leakage model when attacking a device is crucial. In fact, there must be some correspondence between the leakage behaviour of the device and the leakage model for the attack to succeed. An example of this requirement is shown in [19], where CPA is shown to fail if the model is sufficiently wrong. Actually, there is a whole family of countermeasures that try to alter the leakage signal such that it becomes more difficult to model, and thus make the attack harder. One example of a countermeasure belonging to this family is balanced circuits, such as dual-rail logic (for instance WDDL [16]). While in theory dual-rail logic aims at producing data independent leakage, in practice, imperfectly-balanced dual-rail implementations produce a leakage behaviour that is difficult to model, rendering DPA attacks that employ inexact, non-corresponding leakage models ineffective [15] (although in practice regression-based approaches that perform a leakage modeling on the fly [20] can provide good results).

An important enhancement of power analysis are profiled attacks, such as [5] and [14]. These attacks do not require a leakage model previously known to the practitioner, but instead derive the leakage model itself in a profiling step that characterizes the device. The attack step is usually based on Maximum Likelihood classification and can be shown to be optimal if the derived model is sound. During the profiling step, the attacker has virtually unlimited access to a device identical to the device under attack. This access might be difficult to obtain in reality. In short, profiled attacks are a powerful attack strategy that is carried out by a strong adversary.

Recently, several (non-profiled) DPA strategies have been proposed that try to bring the best from both worlds: on the one hand, they do not require a leakage model of the device given beforehand, and thus they could be potentially applied against any device; on the other hand, these generic strategies do not require a profiling step, so that even a weak adversary can perform them. Typically these generic strategies work with measurements in a nominal scale. Exemplary generic DPA strategies are based on Mutual Information [7], on the Kolmogorov-Smirnov distance [17], on the Cramér-von-Mises test [17] or on copulas [18]. We will use MIA in this paper but results apply to all of them.

Despite the attractive properties of generic DPA attacks, earlier papers have identified some theoretical shortcomings of these attacks regarding injective tar-

gets [7,18,19,20]. This paper addresses the practical relevance of these theoretical shortcomings and explores limitations, workarounds and advantages of generic DPA attacks.

Contribution. The contribution of this paper is threefold. Firstly, we present experimental evidence confirming that the "bit drop" trick already proposed in [7] to attack injective targets works even in realistic environments with high, but not infinite, signal-to-noise ratio (SNR). Secondly, we show that by carefully redefining the non-injective targets of the attack, it is indeed possible to launch a fully generic MIA attack on the AES. Thirdly, we demonstrate that precisely the same apparently unappealing property that causes MIA to fail against injective targets enables an attacker to launch MIA attacks exploiting the leakage from an encrypted bus, or from later rounds of AES at reduced number of key hypotheses compared to a traditional DPA attack. The experimental results we provide do not contradict any theoretical result from [18,19,20], but clarify the practical relevance of the issues.

Organization. The remainder of this paper is organised as follows. Section 2 reviews previous work. Section 3 presents some practical experiments regarding the practical applicability of the "bit drop" trick. Section 4 elaborates on a generic MIA attack against the AES. Section 5 studies some properties of MIA that can be exploited to mount MIA attacks on later rounds, and Section 6 concludes the paper.

2 Previous work

In this section we recall the fundamental concepts for MIA attacks and we state the issues that arise when performing MIA attacks on injective targets.

2.1 Notation

Capital letters in bold face, e.g. $\mathbf{X}$, denote random variables. Lower-case letters, e.g. x, denote a specific realization of $\mathbf{X}$, e.g. $\mathbf{X} = x$. The plaintext byte number i is denoted by p_i. The i-th key byte is denoted by k_i. The function drop_i drops i least significant bits of its argument. The expression $I(\mathbf{X}; \mathbf{Y})$ denotes Mutual Information between $\mathbf{X}$ and $\mathbf{Y}$, and is defined [7] as

$$I(\mathbf{X}; \mathbf{Y}) = H(\mathbf{X}) - H(\mathbf{X}|\mathbf{Y}) \tag{1}$$

where $H(\cdot)$ denotes Shannon entropy. S denotes some operation within the computation of a cryptographic primitive. In particular, we denote the AES Sbox by S_{AES}. $\mathbf{L}_{\text{device}}$ denotes the leakage behaviour of the cryptographic device. $\mathbf{L}_{\text{model}}$ denotes a leakage model. Id is the identity function. HW denotes the Hamming weight function. ϵ denotes a Gaussian noise process, its standard deviation is clear from the context.

2.2 Original MIA

MIA was originally introduced in [7] as a new information-theoretic distinguisher that aims at detecting any kind of dependency between observed measurements and predicted power consumption. MIA was different from previous distinguisher flavours in the sense that it is a generic tool suitable for attacking a wide variety of target devices without a priori knowledge, instead of an efficient tool that does require a priori assumptions about the target device. In this respect, MIA neither requires specific knowledge about the exact relation between processed data and observed measurements (leakage behaviour) nor requires to model such behaviour under a leakage model.

More precisely, to target some intermediate value $\mathbf{Z} = S(\mathbf{P} \oplus k)$, MIA uses as distinguisher the value of the Mutual Information between the observed measurements $\mathbf{O} = \mathbf{L}_{\text{device}}(\mathbf{Z})$ and the predicted power consumption $\mathbf{H} = \mathbf{L}_{\text{model}}(\mathbf{Z}_k)$ based on a key guess,

$$I(\mathbf{O}; \mathbf{H}). \tag{2}$$

MIA computes the value of Eq. (2) for each key hypothesis k, ranks the key candidates according to decreasing values of $I(\mathbf{O}; \mathbf{H})$ and selects the key candidate as the one that maximizes the Mutual Information value.

In contrast to other previous distinguishers, such as Pearson's sample correlation coefficient, the Mutual Information value between two random variables expresses the degree of statistical dependency between these two variables, independently of the specific form that the potential dependency between variables may take. This powerful property allows the attacker to skip modelling the leakage behaviour $\mathbf{L}_{\text{device}}$. As a matter of fact, in [7] it was mentioned that the attacker can directly plug in Eq. (2) the hypothesized values handled by the device as the predicted power consumption by setting $\mathbf{H} = \text{Id}(\mathbf{Z})$. By doing so, the attacker does not place any restrictive assumption on the leakage behaviour and thus the attack is expected to work against any device that leaks "somehow".

Moreover, should some information about the leakage behaviour be available beforehand, then that can be used to improve the efficiency of the MIA attack [7], but this is not required for an attack to succeed.

2.3 Issues with injective targets

In several references [7,20] it was noted that for injective targets S (for instance, $S = S_{\text{AES}}(\mathbf{P} \oplus k)$) a straightforward application of Eq. (2) with a generic leakage model by setting $\mathbf{H} = \text{Id}(\mathbf{Z})$ would result in an ineffective attack. Actually, different key hypotheses just imply a permutation of the values $\mathbf{H}$, and thus result in equally meaningful partitions of the observed measurements. This, in turn, leads to the same value for the Mutual Information of Eq. (2) for each key hypothesis, and therefore renders the discrimination of the correct key impossible. Note that this result is not exclusive to MIA, but it extends to a broad set of carefully defined generic attacks, as Whitnall et al. formally reason in [20].

In [7], the authors proposed a fix to solve this issue. They suggested dropping one bit in the predictions, $\mathbf{H} = \text{drop}_1(\mathbf{Z})$, effectively destroying the phenomenon

of different keys causing equivalent (up to a permutation) values of $\mathbf{H}$, yet preserving the generic nature of the attack. This fix is called the "bit drop trick" throughout this paper. They provided empirical evidence of the correctness of this method against one AES Sbox output byte $S = S_{\text{AES}}(p \oplus k)$ with real power measurements using three bits of the prediction, $\mathbf{H} = \text{drop}_5(\mathbf{Z})$.

The bit drop trick has been studied more thoroughly in [18,19]. First, in [18], Veyrat-Charvillon and Standaert provide experiments in a noiseless simulated scenario where the device leaks exactly Hamming weights $\mathbf{L}_{\text{device}}(\mathbf{Z}) = \text{HW}(\mathbf{Z})$ and the target is the AES Sbox output $\mathbf{Z} = S_{\text{AES}}(p \oplus k)$. The predictions consist of the intermediate value $\mathbf{Z}$ when a variable amount of bits are dropped: $\mathbf{H} = \text{drop}_i(\mathbf{Z})$ for $i \in \{1, 2, \ldots, 7\}$. In this situation, they show that although a MIA attack that uses $\mathbf{H} = \text{drop}_i(\mathbf{Z})$ for $i \in \{7, 6, 5, 4, 3, 2\}$ works, a MIA attack that uses $\mathbf{H} = \text{drop}_1(\mathbf{Z})$ unexpectedly fails.

In addition, in [19], Whitnall et al. further study this effect by taking the environmental noise into account with $\mathbf{L}_{\text{device}}(\mathbf{Z}) = \text{HW}(\mathbf{Z}) + \epsilon$. They conclude, from simulations, that a MIA attack on the AES Sbox output using $\mathbf{H} = \text{drop}_1(\mathbf{Z})$ as predictions will fail for large values of the signal-to-noise ratio (that is, when the noise ϵ has small variance). On the other hand, when the signal-to-noise is below a certain threshold, MIA attacks with $\mathbf{H} = \text{drop}_1(\mathbf{Z})$ will eventually succeed. Hence, it was shown that there are high-SNR scenarios where the bit drop trick does not work.

3 Practical relevance of the bit drop trick

In this section, we experimentally verify the practical relevance of the negative results regarding dropping some bits in the predictions.

We performed all the experiments for this paper on an 8-bit micro controller from Atmel's AVR family running an unprotected implementation of AES-128. We obtained $10,000$ power traces from encryptions of randomly chosen plaintexts covering the first one and a half rounds. The card is clocked at 4 MHz and the sampling frequency is 10 MS/s. The device leakage behaviour is known to be close to Hamming weight, however, in what follows we do not make use of this fact and proceed as if the leakage behaviour were unknown to us.

3.1 Dropping one bit

We performed a MIA attack targeting the first Sbox output, by setting as targeted intermediate value the byte $\mathbf{Z} = S_{\text{AES}}(\mathbf{P}_1 \oplus k_1)$ and dropping one bit on the predictions $\mathbf{H} = \text{drop}_1(\mathbf{Z})$. The (non-parametric) density estimation process was performed without placing any hypotheses on the leakage behaviour of the device. Thus, densities were estimated with histograms with 256 bins, since measurement samples have a resolution of 8 bits.

Figure 1 shows the result this attack when using $1,000$ traces at time sample 485. It can be observed that the attack is successful, the correct key clearly stands out over all other competing key hypotheses. We confirmed that around

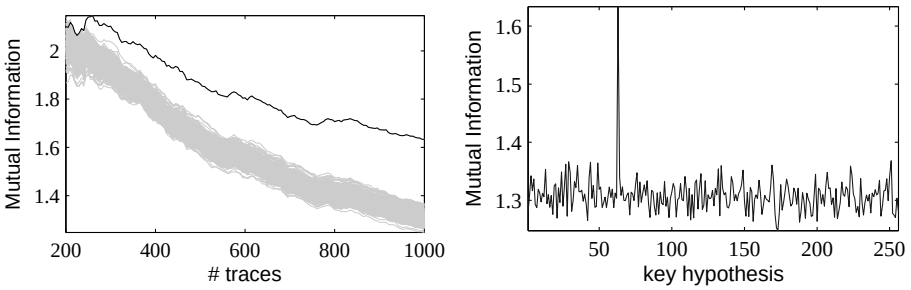

Fig. 1. Left: evolution of the MIA attack targeting the intermediate value $z = S_{\text{AES}}(p \oplus k)$ dropping one bit in the predictions, $\mathbf{H} = \text{drop}_1(\mathbf{Z})$, as the number of traces increases. Correct key in black, incorrect key hypothesis in grey. Right: outcome of the attack using $1,000$ traces.

time sample 485 the implementation performs the Sbox lookups from the first round.

The experiment highlights the limited impact of the negative results in [19,18] regarding the drop_1 trick. We can hardly conceive a practical scenario with a higher SNR than ours, since our platform (AVR) has relatively high leakage for industry standards. Furthermore, in reality implementations will feature some countermeasures that will only degrade the SNR. Thus, the negative results of [19] regarding the bit drop trick in strong-signal settings are of theoretical relevance but we conclude that the bit drop trick works in practical, high yet finite SNR scenarios. In other words, the high SNR of our measurements is "low enough" for the bit drop trick to work.

3.2 Dropping more than one bit

The experiment from Section 3.1 was repeated for a different amount of least significant bits dropped, ranging from 1 to 7, following the spirit of [18]. All the other parameters of the attack are kept constant: we focus on the same time sample 485 and use the same Mutual Information estimation procedure with histograms of 256 bins in order not to place any assumption about the leakage behaviour during the density estimation process.

Figure 2, left, shows 7 plots for 7 different attacks. Each attack drops a different number of bits (leftmost drops 1 bit, rightmost drops 7 bits). Each subplot shows the evolution of the corresponding attack as the number of traces increases.

The attacks are, in every case, successful. Note that the mutual information decreases as the number of bits dropped increases. This is natural and can be easily explained by expanding Eq. 2 as Eq. 1 and noting that $H(\mathbf{O}|\mathbf{H})$ will only increase as $\mathbf{H}$ considers fewer bits of $\mathbf{Z}$.

Albeit the attacks are eventually successful for any number of dropped bits, the distinguishing capabilities are not the same. Figure 2, right, studies the effect

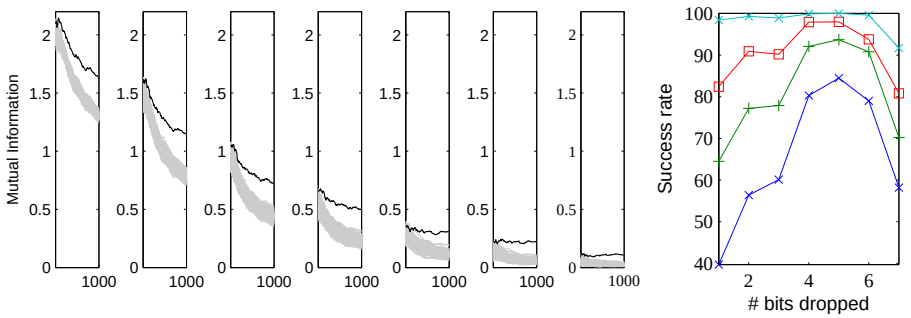

Fig. 2. Left: evolution of the performance of the attack from Figure 1 for 1, 2, ..., 7 dropped bits. For each sub plot the X axis is number of traces and ranges from 200 to 1000. Correct key hypothesis is in black, incorrect key hypotheses are in grey. Right: success rate of the same attacks as in left. Each line represents a fixed number of traces. Bottommost line is 200 curves, next lines correspond to increments of 40 traces.

of different number of bits dropped on the success rate. Each line corresponds to a fixed amount of power traces, starting with 200 curves for the line at the bottom. All the parameters in the estimation process (number of bins) were kept constant, allowing for a fair comparison.

We can see two main tendencies in Figure 2. First, as the number of dropped bits increases, from 1 to 5, the success rate also increases, for a given number of traces. Second, if the number of dropped bits continues increasing to 6 and 7, the success rate drops. Thus, for this particular target and this particular attack, the optimum number of bits to drop to maximize the success rate of this attack is 5. We believe that this fact is the result of the superposition of two opposite effects. As the number of dropped bits increases, the target becomes "more non-injective" and thus the attack works better. This is true until a certain threshold (in our case, 5 bits), from where the effect of the algorithmic noise introduced by the dropped, unmodelled bits is predominant. Note that this observation fits nicely with the classic result of Messerges in [11,12], who found that in the context of d-bit DPA, modelling only 3 bits from an 8-bit bus gave the best SNR for a fixed number of traces.

4 Non-injective targets on AES

There are situations, however, in which an attacker might not be able to use the bit drop trick. As pointed out in the experiments in [19], if the leakage behaviour of the device consists only of higher order terms, the bit drop trick will fail. In this section, we note that an attacker who wishes to recover the key from an AES implementation is not restricted to only performing attacks against the Sbox output. Thus, he is not forced to employ the bit drop trick. Although we focus on the AES, we expect that any reasonable block cipher contains non-injective target functions.

In the rest of this section, we give an example of such a non-injective target that enables generic MIA attacks. This target was briefly mentioned in [1], in the following we provide a detailed study and show its attractive properties.

4.1 Suitable targets

There are suitable non-injective targets for an attack in AES. Intuitively speaking, we want to find some intermediate value $\mathbf{Z}$ that results from the *compression* of public and secret data. One such value $\mathbf{Z}$ suitable for generic MIA attacks naturally arises as follows.

Let us take a closer look at the computation of the MixColumns transformation that is applied to the first column of the state during the first round. Name (u, v, w, x) the 4-byte input column to this MixColumns transformation. These 4 input bytes correspond to 4 output bytes of ShiftRows in the first round. Suppose that during the computation of the first output byte $2u \oplus 3v \oplus w \oplus x$ of this MixColumns invocation, the implementation handles the partial intermediate value $z = 2u \oplus 3v = 2S_{\text{AES}}(p_1 \oplus k_1) \oplus 3S_{\text{AES}}(p_6 \oplus k_6)$[1]. The target $z = 2u \oplus 3v$ is obviously non-injective (since it maps 16 bits of public input and 16 bits of secret input to 8 bits) and suitable for a generic MIA attack. Below we provide results of such an attack against our unprotected AES software implementation.

4.2 Practical results

Since we are attacking a non-injective target, we can use directly $\mathbf{Z}$ as the predicted power consumption (without the bit drop trick), and proceed with a generic MIA attack on the 16 key bits (corresponding to key bytes 1 and 6). That is, we use the identity leakage model $\mathbf{H} = \text{Id}(\mathbf{Z}) = \mathbf{Z}$.

To perform the attack without placing any assumption on the leakage behaviour, Mutual Information was estimated using histograms with 256 bins. In Figure 3 we can see that the attack works correctly: the correct key stands out and the attack has a unique solution. Note also that the magnitude of the Mutual Information values is different than in the previous section, this is because we are targeting a different time sample, 523, corresponding to the computation of MixColumns.

4.3 Different leakage behaviours

The bit drop trick is known not to work against devices with highly non-linear leakage behaviours. However, a generic MIA attack against a non-injective target does not suffer from this limitation and can be used in such cases.

Unfortunately, we do not have access to a chip with highly non-linear leakage behaviour. For this reason, we resort to simulations to give an example of an attack against $z = 2u \oplus 3v$ on an hypothetical device that leaks as

[1] Note that the sixth Sbox output of the first round will be in state byte 5 after first round ShiftRows.

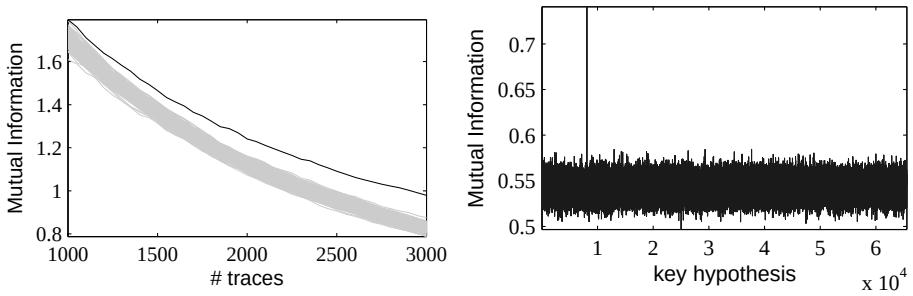

Fig. 3. Left: evolution of the generic MIA attack targeting the intermediate value $z = 2S_{\text{AES}}(p_1 \oplus k_1) \oplus 3S_{\text{AES}}(p_6 \oplus k_6)$. Correct key in black, incorrect keys in grey. Time sample index 523. Right: outcome of the attack using $3,000$ traces.

$\mathbf{L}_{\text{device}}(\mathbf{Z}) = S_{\text{AES}}(\mathbf{Z})$ (resembling a highly non-linear leakage behaviour) and as $\mathbf{L}_{\text{device}}(\mathbf{Z}) = \text{HW}(S_{\text{AES}}(\mathbf{Z}))$ (resembling an encrypted bus scenario as explained in the following section).

Figure 4 shows the outcomes of these attacks when the simulated curves include Gaussian additive noise with standard deviation $\sigma = 1$. The attacks are successful in both cases $\mathbf{L}_{\text{device}}(\mathbf{Z}) = S_{\text{AES}}(\mathbf{Z})$ and $\mathbf{L}_{\text{device}}(\mathbf{Z}) = \text{HW}(S_{\text{AES}}(\mathbf{Z}))$. Mutual Information was estimated, as in previous cases, using histograms and 256 bins. The attacks are successful also when there is no noise $\sigma = 0$ in the simulated traces.

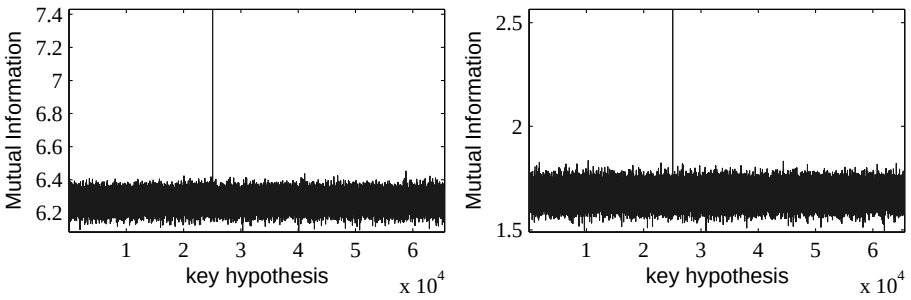

Fig. 4. Left: Simulation of the attack from Section under leakage behaviour $\mathbf{L}_{\text{device}} = S_{\text{AES}}$. Right: idem, under $\mathbf{L}_{\text{device}} = \text{HW}(S_{\text{AES}})$

To sum up, unlike the drop_1 strategy on the AES Sbox output, generic MIA attacks against non-injective target functions work under any leakage condition, any SNR level at the expense of a slightly more costly computational effort of 2^{16} key hypotheses.

5 Discussion

In this section we elaborate on the notable properties of the previous attack that are of practical relevance when attacking devices with encrypted buses. In addition, we comment on the possibility of mounting MIA attacks on later rounds and discuss the implications of a parallel implementation of the AES.

5.1 Arbitrary leakage model and bus encryption

As already stated in Section 4.3, generic MIA attacks against non-injective targets in AES will work with any leakage behaviour. This fact has important consequences for real-world devices that employ bus encryption (also called *bus scrambling*.) We can successfully apply the attack of Section 4 even if the attacker has only access to the leakage of some "encrypted" version of the intermediate value (under a constant key), instead of the leakage of the actual intermediate value. An advantage of this attack is that the adversary does not need to know the exact details of the "encryption" function (for example, the key might be unknown). The attacker launches the generic attack of Section 4 using as predictions the "unencrypted" values for the intermediate sensitive variable, disregarding the exact details of the encryption function. In what follows we explain why this attack returns the correct key.

For instance, consider a smart-card that "encrypts" the values before sending them over its bus[2]. Let $\mathbf{Y}$ be the "encrypted" value of $\mathbf{Z}$, that is, $\mathbf{Y} = \mathrm{enc}(\mathbf{Z})$ for some bus "encryption" function enc (which is a permutation). We claim that the following two attacks are equivalent (that is, have equal success rate):

1. A MIA attack against a device that handles the values in clear, $\mathbf{O} = \mathbf{L}_{\mathrm{device}}(\mathbf{Z})$ and the attacker predictions correspond to the values in clear, $\mathbf{H} = \mathbf{Z}$.
2. A MIA attack against a device that handles the intermediate values encrypted, $\mathbf{O} = \mathbf{L}_{\mathrm{device}}(\mathbf{Y})$ and the attacker uses as predictions the values in clear, $\mathbf{H} = \mathbf{Z}$.

We will show that both attacks produce the same values of mutual information, $I(\mathbf{L}_{\mathrm{device}}(\mathbf{Y}); \mathbf{L}_{\mathrm{model}}(\mathbf{Z})) = I(\mathbf{L}_{\mathrm{device}}(\mathbf{Z}); \mathbf{L}_{\mathrm{model}}(\mathbf{Z}))$, and thus have the same behaviour. We can develop Eq. (2) as

$$\begin{aligned} I(\mathbf{L}_{\mathrm{device}}(\mathbf{Y}); \mathbf{L}_{\mathrm{model}}(\mathbf{Z})) &= I(\mathbf{L}_{\mathrm{device}}(\mathbf{Y}); \mathbf{L}_{\mathrm{model}}(\mathbf{Y})) \\ &= I(\mathbf{L}_{\mathrm{device}}(\mathbf{Z}); \mathbf{L}_{\mathrm{model}}(\mathbf{Z})) \end{aligned} \tag{3}$$

where the first line results from $\mathbf{Y}$ and $\mathbf{Z}$ being related by a permutation (note that the attack from Section 4 uses $\mathbf{L}_{\mathrm{model}} = \mathrm{Id}$) and the second line results from $\mathbf{Y}$ and $\mathbf{Z}$ being identically (uniformly) distributed (thus, due to symmetry

[2] Here we are assuming without loss of generality that the most leaking component of the smart-card is the bus. This assumption is just for illustration purposes and the following discussion is orthogonal to this assumption.

of H, the value of $H(\mathbf{Z}) - H(\mathbf{Z}|\mathbf{L}_{\text{device}}(\mathbf{Z}))$ does not change if the events of $\mathbf{Z}$ are reordered).

Therefore, the previous result shows that the bus encryption has no effect on generic DPA attacks and is transparent to them. The attacker can recover the key observing leakage of the encrypted intermediate values, even if he ignores the exact details of the bus encryption mechanism and the exact leakage behaviour of the device.

5.2 Arbitrary leakage model and absorbing next round keys

The very same property used in the previous Subsection 5.1 can be exploited to mount MIA attacks that use observations of intermediate values from the second round at a reduced number of key hypotheses. The rationale is similar: Suppose that the attacker chooses one output byte of MixColumns as the target, $z'' = 2u \oplus 3v \oplus w \oplus x$. Assume that the value z'' is later transformed into z''' through AddRoundKey and further into z^{IV} through SubBytes of the second round, $z''' = \text{AK}(z'')$ and $z^{IV} = \text{SB}(z''')$ as shown in Figure 5. These operations merely permute z'' (they do not introduce any other "variable" data), and thus the result from Subsection 5.1 can be applied. The attacker can perform the attack placing hypotheses on $\mathbf{Z}''$ as $\mathbf{L}_{\text{model}}(\mathbf{Z}'') = \text{Id}(\mathbf{Z}'')$ but he can use the observations $\mathbf{O}$ corresponding to the handling of the intermediate value from the second round such as z^{IV} with a 2^{32} work load. This is not possible with standard DPA, unless the attacker places 8 additional bits on the key hypothesis corresponding to one key byte from the second round, resulting in an attack with 2^{40} work load.

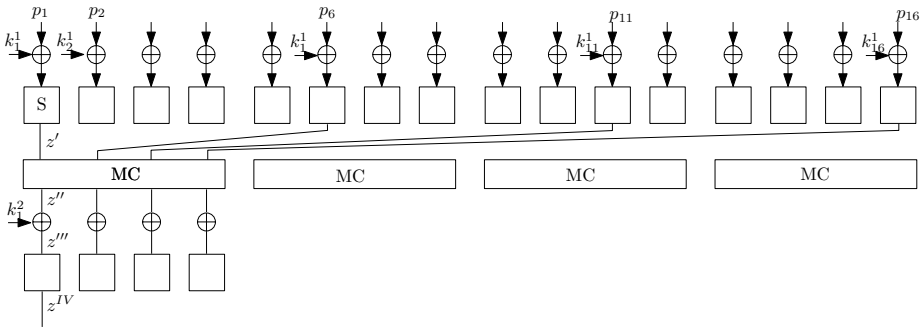

Fig. 5. A schematic representation of the computation of the first 1.5 rounds of AES. As explained in Section 5.2, the attacker can use the leakage coming from later in the computation, for instance, z''' after the next key addition or even z^{IV} after the Sbox lookup, without placing extra key hypotheses on the second round key k_1^2.

This means that the attacker can easily bypass the protections in the first round and exploit directly leakage from the second round to recover first round

keys. This is helpful for the attacker if only the first round is protected with, for example, the masking countermeasure and highlights, again, the importance of not only masking the outer rounds, as e.g. discussed in [3].

5.3 What happens if MixColumns leaks in parallel?

In Section 4, we assumed that the implementation computes MixColumns in a serial fashion. In this section, we analyse the case when this no longer holds, i.e., the implementation computes and leaks in parallel (for example, a hardware implementation using a 32-bit or 128-bit data path). In what follows we will work with a reduced exemplary data path of 16 bits (2 bytes). (The implications of the study can be extrapolated to any other bit width.)

Suppose that the observations $\mathbf{O}$ available to the attacker correspond to the simultaneous leakage $\mathbf{O} = \mathbf{L}_{\text{device}}(\mathbf{Z}, \mathbf{N})$ of two output bytes z and n from first round MixColumns. Suppose the attacker sets z as target (and therefore n will be considered as "algorithmic noise") and executes the attack of Section 4. We can distinguish two cases:

1. The leakage contribution of $\mathbf{Z}$ can be "decoupled" from that of $\mathbf{N}$. For instance, $\mathbf{L}_{\text{device}}(\mathbf{Z}, \mathbf{N}) = \text{HW}(\mathbf{Z}) + \text{HW}(\mathbf{N})$. Here the attacker can eventually cancel the algorithmic noise induced by the term $\text{HW}(\mathbf{N})$ (note that $\mathbf{Z}$ and $\mathbf{N}$ are statistically independent) and the attack will succeed. Another example of $\mathbf{L}_{\text{device}}$ that falls in this category is $\mathbf{L}_{\text{device}}(\mathbf{Z}, \mathbf{N}) = \text{HW}(S_{\text{AES}}(\mathbf{Z})) + \text{HW}(S_{\text{AES}}(\mathbf{N}))$.
2. The leakage contribution of $\mathbf{Z}$ cannot be "decoupled" from that of $\mathbf{N}$. For instance, $\mathbf{L}_{\text{device}}(\mathbf{Z}, \mathbf{N}) = \text{HW}(\mathbf{Z} \oplus \mathbf{N})$. Here the attacker cannot cancel the algorithmic noise (the signal from $\mathbf{Z}$ is effectively "masked" with $\mathbf{N}$, which by assumption is unknown to the attacker). Hence, the attack of Section 4 will not work. Another example that falls in this category is $\mathbf{L}_{\text{device}}(\mathbf{Z}, \mathbf{N}) = \text{HW}(\text{SHA}(\mathbf{Z}||\mathbf{N}))$ where $||$ is concatenation and SHA is a cryptographic hash function.

The requirement for a leakage behaviour to belong to the first category is that the distribution of $\mathbf{L}_{\text{device}}(\mathbf{Z}, \mathbf{N})$ should still be informative about $\mathbf{Z}$ when the effect of $\mathbf{N}$ is marginalized, that is,

$$I(\mathbf{Z}; \mathbb{E}_{\mathbf{N}}[\mathbf{L}_{\text{device}}(\mathbf{Z}, \mathbf{N})]) > 0. \tag{4}$$

In fact we can also use the example of a MIA attack against an injective Sbox to illustrate the restriction of Eq. 4. We can rewrite $\mathbf{L}_{\text{device}}(\mathbf{Z}, \mathbf{N}) = L_0(L_1(\mathbf{Z}), L_2(\mathbf{N}))$ where $\mathbf{Z}$ are 7 bits of the Sbox output and $\mathbf{N}$ is the other bit of the Sbox output. If L_0 is such that Eq. 4 holds, the trick of using drop_1 will work. On the other hand, if L_0 is such that Eq. 4 does not hold, the attack will fail.

Hence, in the case of a parallel MixColumns implementation, if L_0 is such that Eq. 4 does not hold, we would have to redefine the target value as $\mathbf{Z}' = (\mathbf{Z}, \mathbf{N})$, and we find ourselves in the same situation as if we were attacking a (large)

injective Sbox. Hence, we would need to choose another suitable non-injective target, deeper in the algorithm, at the cost of more key hypotheses.

What does it mean to impose constraints on L_0? We point out that these restrictions on L_0 are easily met in practice. For example, the restrictions mean that we can allow arbitrary cross-talk between the wires that represent $\mathbf{Z}$, and also arbitrary cross-talk between the wires that represent $\mathbf{N}$, but there should not be only significant cross-talk between wires from both variables. Or, equivalently, leakage of $\mathbf{Z}$ can be "encrypted", and also that of $\mathbf{N}$, but not jointly "encrypted". Intuitively, we allow arbitrary leakage of $\mathbf{Z}$ and of $\mathbf{N}$, but impose some mild restrictions on how these individual leakages are combined (through L_0).

6 Conclusion

In this work, we elaborated on the practical properties of the bit drop trick, pointed out how generic DPA attacks can be mounted on the AES and showed their appealing properties when attacking devices with encrypted leakage or exploiting leakage from inner rounds. Echoing the title, generic attacks are certainly endowed with two-sided properties - curse or blessing depending on the concrete situation.

Acknowledgments. We thank the anonymous reviewers for their thorough evaluation. This work was supported in part by the Research Council of KU Leuven: GOA TENSE (GOA/11/007), by the Flemish Government FWO G.0550.12N and by the Hercules Foundation AKUL/11/19. Oscar Reparaz is funded by a PhD Fellowship of the Fund for Scientific Research - Flanders (FWO). Benedikt Gierlichs is Postdoctoral Fellow of the Fund for Scientific Research - Flanders (FWO).

References

1. L. Batina, B. Gierlichs, E. Prouff, M. Rivain, F.-X. Standaert, and N. Veyrat-Charvillon. Mutual information analysis: a comprehensive study. *J. Cryptology*, 24:269–291, 2011.
2. R. Bevan and E. Knudsen. Ways to enhance differential power analysis. In P. Lee and C. Lim, editors, *Information Security and Cryptology ICISC 2002*, volume 2587 of *Lecture Notes in Computer Science*, pages 327–342. Springer Berlin Heidelberg, 2003.
3. A. Biryukov and D. Khovratovich. Two new techniques of side-channel cryptanalysis. In P. Paillier and I. Verbauwhede, editors, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 195–208. Springer, 2007.
4. E. Brier, C. Clavier, and F. Olivier. Correlation power analysis with a leakage model. In M. Joye and J.-J. Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer Berlin Heidelberg, 2004.

5. S. Chari, J. Rao, and P. Rohatgi. Template attacks. In B. Kaliski, C. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer Berlin Heidelberg, 2003.
6. J.-S. Coron, P. C. Kocher, and D. Naccache. Statistics and secret leakage. In *Proceedings of the 4th International Conference on Financial Cryptography*, FC '00, pages 157–173, London, UK, UK, 2001. Springer-Verlag.
7. B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel. Mutual Information Analysis - A Generic Side-Channel Distinguisher. In E. Oswald and P. Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008*, volume 5154 of *Lecture Notes in Computer Science*, pages 426–442, Washington DC,US, 2008. Springer-Verlag.
8. P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In N. Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 104–113. Springer, 1996.
9. P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.
10. R. Mayer-Sommer. Smartly analyzing the simplicity and the power of simple power analysis on smartcards. In *Proceedings of the Second International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '00, pages 78–92, London, UK, UK, 2000. Springer-Verlag.
11. T. S. Messerges, E. A. Dabbish, and R. H. Sloan. Investigations of power analysis attacks on smartcards. In *Proceedings of the USENIX Workshop on Smartcard Technology on USENIX Workshop on Smartcard Technology*, WOST'99, pages 17–17, Berkeley, CA, USA, 1999. USENIX Association.
12. T. S. Messerges, E. A. Dabbish, R. H. Sloan, and S. Member. Examining smartcard security under the threat of power analysis attacks. *IEEE Transactions on Computers*, 51:541–552, 2002.
13. E. Oswald. *On side-channel attacks and the application of algorithmic countermeasures*. PhD thesis, Graz University of Technology, 2003.
14. W. Schindler, K. Lemke, and C. Paar. A stochastic model for differential side channel cryptanalysis. In J. R. Rao and B. Sunar, editors, *Cryptographic Hardware and Embedded Systems CHES 2005*, volume 3659 of *Lecture Notes in Computer Science*, pages 30–46. Springer Berlin Heidelberg, 2005.
15. K. Tiri, D. Hwang, A. Hodjat, B.-C. Lai, S. Yang, P. Schaumont, and I. Verbauwhede. Prototype IC with WDDL and differential routing - DPA resistance assessment. In *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 354–365. Springer, 2005.
16. K. Tiri, D. D. Hwang, A. Hodjat, B.-C. Lai, S. Yang, P. Schaumont, and I. Verbauwhede. Prototype IC with WDDL and differential routing - DPA resistance assessment. In J. R. Rao and B. Sunar, editors, *CHES*, volume 3659 of *Lecture Notes in Computer Science*, pages 354–365. Springer, 2005.
17. N. Veyrat-Charvillon and F.-X. Standaert. Mutual information analysis: How, when and why? In C. Clavier and K. Gaj, editors, *CHES*, volume 5747 of *Lecture Notes in Computer Science*, pages 429–443. Springer, 2009.
18. N. Veyrat-Charvillon and F.-X. Standaert. Generic side-channel distinguishers: Improvements and limitations. In P. Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 354–372. Springer, 2011.

19. C. Whitnall and E. Oswald. A fair evaluation framework for comparing side-channel distinguishers. *Journal of Cryptographic Engineering*, 1(2):145–160, August 2011.
20. C. Whitnall, E. Oswald, and F.-X. Standaert. The myth of generic DPA...and the magic of learning. CT-RSA 2014, Lecture Notes in Computer Science, vol 8366, pp 183-205, San Fransisco, USA, February 2014, 2012. `http://eprint.iacr.org/`.